E Pluribus Deanonymization: Fingerprinting Browsing Sessions Instead of Individual Webpages

Nathan S. Pan, Phan Nguyen, Joshua Wang, Dave Levin

Abstract

Website fingerprinting attacks on Tor seek to identify the specific webpage visited by a user given only the anonymized traffic to and from that user. Such attacks have been shown to be effective when the set of webpages a user visits can come from one of only a small set of candidates, but become significantly less effective as the set of candidates grows large.

In this paper, we demonstrate a way to significantly improve the performance of virtually all website fingerprinting techniques to work on a large set of candidates. Our insight is that users very rarely ever visit a single website, but rather visit multiple, related websites in succession over the course of a browsing session. Thus, rather than try to fingerprint a single website independent of all others, we fingerprint the browsing session itself, using consecutive websites as contextual clues.

We introduce a browsing session fingerprinting technique that uses as a building block any (individual-)website fingerprinting technique. We evaluate it using the largest public fingerprinting dataset to date, which we collected, and show that our techniques can achieve a top-1 accuracy that approaches the individual-website fingerprinting techniques' top-5 accuracy. We will be making our code and data publicly available.

1 Introduction

Every day, millions of people use Tor [8] to anonymously browse the web and communicate [26]. More precisely, Tor provides a form of anonymity known as *unlinkability*, which permits an attacker to know at most one side of a communicating pair (e.g., either the client or the server), but not both. Ensuring Tor's resilience to deanonymization attacks is critical in helping users to evade threats to Internet freedom.

Some of the most pernicious attacks on Tor's anonymity are *website fingerprinting attacks*. The insight behind these attacks is that, while the actual content (and, of course, destination) of Tor packets are hidden, the sequence of packet uploads and downloads themselves can serve as a *fingerprint* of the specific webpage being downloaded. The attack thus involves the adversary first visiting a set of candidate webpages and collecting the resulting fingerprints. The attacker then observes a target user's traffic *while they visit a single webpage*, and compares the traffic against the collected fingerprints to determine which webpage they are visiting—thereby violating unlinkability.

While simple in principle, website fingerprinting attacks in practice remain extremely challenging. This is because they have to contend with network randomness and dynamic webpage content, both of which result in nondeterministic fingerprints for even simple webpages. As a result, today's website fingerprinting attacks involve increasingly sophisticated machine learning (ML) techniques, yet still have significant errors, even when the attacker has only a modest number of candidate webpages. For example, we evaluate Triplet Fingerprinting (TF) [25], a state-of-the-art ML-based website fingerprinting attack on a candidate set of 7,110 webpages and find that it determines the correct webpage precisely only 58% of the time, and places it within its top-5 guesses 77% of the time.

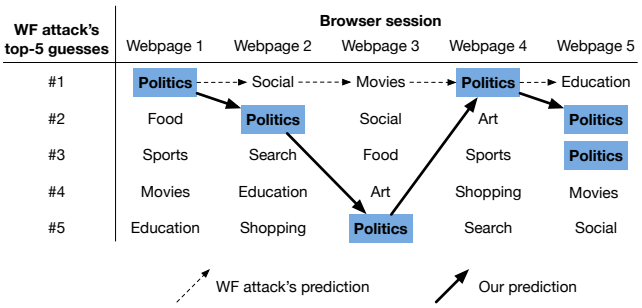


Figure 1: Overview of our browser session fingerprinting attack. Rather than simply return each top-1 guess, we identify common trends amongst the top-5 as contextual clues for what pages the user may have visited. In this example, the user was visiting politics-related webpages.

In this paper, we introduce a fundamentally new approach to website fingerprinting: instead of viewing and trying to identify only a *single* webpage that the target user is visiting, the attacker is able to view *multiple consecutive* webpages visited by the user—what we refer to as a "browsing session"—and seeks to identify as many of them as possible. We believe this attack to be more realistic than website fingerprinting attacks—after all, it is rare for a user to start up their browser, visit a single webpage, and then shut down their browser.

Moreover, we demonstrate that, under reasonable assumptions, this attack is also more *dangerous* than traditional website fingerprinting. To understand our insight, consider an attacker monitoring each webpage a user visits, and applying a website fingerprinting technique to each.

Our underlying assumption is that webpages visited consecutively by a user are likely (but not guaranteed) to be related to one another—such as viewing multiple news webpages, several different restaurant-related pages, or various message boards on the same social media website. When this reasonable assumption holds, we show that the attacker can use consecutively-visited webpages as contextual cues to achieve better accuracy.

Figure 1 provides an example of our attack. The attacker uses a standard website fingerprinting tool to obtain its top-5 guesses for each webpage the user has visited. Then, using information about

Submitted to ACM CCS 2025, Taipei, Taiwan
2025. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

the relatedness of the different candidate webpages, the attacker looks for common trends amongst them, and infers what amounts to the context of the user's browsing at that point in time. In this example, either the user is randomly jumping across multiple contexts, or the user has visited a series of politics-related webpages—we argue the latter is more likely. Using this additional information allows the attacker to more confidently guess webpages that were not already selected in the top-1 spot. We formalize and thoroughly evaluate this approach in this paper.

Contributions This paper makes the following contributions:

- We introduce browser session fingerprinting, a generalization of website fingerprinting that leverages contextual clues from consecutively visited webpages.
- We introduce a new, controlled website fingerprinting dataset, the largest of its kind to date.
- We present the first browser session fingerprinting attack and evaluate it on our dataset and others', demonstrating that, under reasonable assumptions, it is able to drastically improve the accuracy of identifying individual webpages.
- We will be making our code and dataset publicly available.[1]

Roadmap The rest of this paper is structured as follows. We review background and related work in §2. We present our attack's design in §3, and our new dataset in §4. We perform a thorough evaluation of our attack in §5, and conclude in §6.

2 Background and Related Work

In this section, we review relevant details of Tor, website fingerprinting attacks, and fingerprint datasets.

2.1 Tor background

Tor [8] is a peer-to-peer, anonymizing communication network. Participants can join the Tor network and run a *relay* to help forward traffic for users. A client creates a Tor *circuit* comprising (at least) three Tor relays: an entry node, a middle node, and an exit node. Circuits are constructed in such a manner that each hop only learns the identity of the hops immediately preceding and succeeding it.

To mitigate information leaked from the packet sizes, Tor segments all of its communication into fixed-sized cells, and communicates through encrypted TLS sessions. In practice, TLS can choose to combine multiple separate application packets into a single record, and TCP can combine or segment data, so while every individual on-the-wire packet may not be the same size, the raw amount of information is.

Research into attacking [9, 31] defending [14, 16, 30], and optimizing [1, 3, 17] Tor is rich and varied, but we focus in this paper on improving a particular kind of attack: website fingerprinting.

2.2 Website fingerprinting attacks and defenses

Website fingerprinting (WF) attacks involve an attacker who can view a Tor user's traffic, exploiting the (semi-)deterministic nature of how websites' data appears on the wire to construct "fingerprints" of webpages. WF attacks [4, 7, 10, 11, 18, 22, 24, 25, 27, 28] are possible in Tor because, traditionally, it does not inject "junk" traffic, unlike mixnets [6]. Nonetheless, there is some non-determinism in website downloads, due to varying network conditions and dynamic web content. To overcome these challenges, WF attacks have evolved to incorporate increasingly sophisticated machine learning techniques [25].

Of particular relevance to this work is how WF attacks are evaluated. Attackers operating in a "closed world" model are assumed to have identified some number N of candidate webpages that the victim user may attempt to visit, and the goal is to determine which of the N they did. Accuracy is typically measured in one of two ways: the *top-1* accuracy is the fraction of the time the WF technique can guess the precise webpage that the user went to. The *top-5* (or more generally top-k) accuracy is the fraction of time that the actual website is among the WF technique's top five guesses. Naturally, as N increases, the accuracy tends to decrease.

Defenses to WF attacks [4, 5, 15, 29] involve reordering and batching requests, injecting junk traffic, and offloading the download to a programmable agent [21].

Despite these impressive defensive efforts, WF attacks are still able to achieve high degrees of accuracy, though are still imperfect. Cherubin et al. [7] showed that Triplet Fingerprinting (TF) [25]—a state of the art ML-based technique—quickly degrades in accuracy with even modest increases to the size of the closed world. We show in §4 that TF is able to guess the precise webpage 46–64% of the time, and can guess the website within the top-5 75–81% of the time, depending on the fingerprinting dataset and size of the closed world.

We believe that our work significantly extends this large body of prior work by introducing a new, orthogonal dimension to website fingerprinting. Our work is, to the best of our knowledge, the first to take into account the fact that users rarely visit webpages purely at random, and that one can use information gleaned from one of a user's webpage visits to help identify another. The attack that we present in §3 uses existing WF techniques as a core building block, thus as WF techniques improve, so too does our attack. Moreover, we show that our attack can drastically improve the accuracy of even poorly-performing WF techniques. Thus, we believe that future WF work should evaluate based not only on guessing individual webpages, but on identifying browser sessions as performed in this paper.

2.3 Fingerprint datasets

One of our major contributions in this work is the curation and public release of a new fingerprint dataset. Fingerprint datasets are critical to supporting research into website fingerprinting, as they are used to train and evaluate attack models.

There are three critical features that a fingerprint dataset should have. *First*, it should span a large number of individual domains, so as to allow researchers to test large closed-world attacks. *Second*, it should comprise not just landing pages (the webpage when the URL is simply the domain name), but also *internal pages* to that domain. Internal pages are where users spend most of their time browsing, and they have been shown to be fundamentally different from landing pages [2] in terms of content, size, performance optimizations,

[1] We do not yet make them publicly available, as the size of our dataset has made it challenging to post it somewhere where we can retain author anonymity.

and even adoption of HTTPS [19]. *Finally*, the pages visited should approximate real human behavior to the extent possible.

Several such datasets have been introduced [22, 24, 27], but of particular note are the GoodEnough (GE) [20] and BigEnough (BE) [15] datasets. Table 1 in §4 compares these datasets to the one we collect.

Recently, Cherubin et al. [7] evaluated a new method of obtaining fingerprint data: instead of obtaining synthetic website visits, they collected data to and from exit relays they ran. Their evaluation showed that, even though it is collected at the exit and not at the entry (where website fingerprinting attacks take place), the dataset loses virtually no fidelity as a tool for training a website fingerprinting classifier. They have since released their datasets publicly [10]. Such a dataset has the benefit that it is driven by real user behavior, and is significantly larger than other synthetic datasets by several orders of magnitude. However, because they collect data at the exit relays, they can only see the (ostensible) domain in the TLS SNI, and cannot determine what the precise internal page was. We felt this was important information for our experiments, so we constructed our own synthetic dataset.

3 Attack Design

Here, we present the intuition, design, and limitations of our browser session fingerprinting attack.

3.1 Intuition

Users rarely open up a browser, go to a single webpage, and then close the browser. Yet this is essentially how today's website fingerprinting attacks model this problem: each webpage being visited in complete isolation of any other.

Rather, users typically navigate to multiple pages, often with related pages viewed in direct succession. For example, a user looking for information about a sensitive political topic might view historical information about that topic, read the latest news about it, and engage in several online forms to discuss the current events. In other words, subsets of a user's browser session span a particular "context" of webpages.

The insight behind browser session fingerprinting attacks is that, if the attacker can infer the current context the user is in, they can use this to help refine their accuracy.

Most website fingerprinting techniques today do not just output a single guess of a webpage, but rather assign probabilities to *each* webpage. Indeed, it is standard for website fingerprinting papers to report on both their top-1 accuracy (the probability of guessing the correct webpage exactly) as well as their top-5 accuracy (the probability that one of their top-5 guesses is the correct webpage. Naturally, an attack's top-5 accuracy is better than its top-1; for example, we show in §4.3 that Triplet Fingerprinting [25] is able to obtain a top-1 accuracy of 58% and a top-5 of 77%, when applied to a fingerprint dataset we collected.

3.2 Model of user browsing behavior

The core assumption underlying our attack is that users' browsing behavior can be modeled in a very crude manner. Specifically, we assume that it is possible to separate webpages into groups of related webpages, such that if a user is on a webpage from group G, then the user is more likely to subsequently visit a webpage that is also in G than to switch to another group. Put another way, we assume that the conditional probability of visiting a webpage in group G given that the previously visited webpage was also in G ($\Pr[G|G]$) is greater than if it had come from an unrelated group ($\Pr[G|H \neq G]$). This is in essence the intuition behind Page Rank; users are likely to navigate in large part by clicking on links they find on the current webpage they are on, and those links are likely to be on the same topic.

There are many possible ways to group webpages together, such as based on the topic of their content, whether they have hyperlinks to one another, or by collecting users' browsing habits to construct a predictive model. We do not claim to have an accurate model of user behavior; it is an interesting and important area of future work, but we consider it out of scope for this paper. However, to demonstrate our sensitivity to the quality of the user model, we evaluate in §5 against a set of worst-case models, and still show our ability to achieve improved accuracy.

User model used in this paper The specific model of users' browsing behavior we consider here proceeds as follows. We cluster the webpages into non-overlapping groups—we evaluate several different groupings, including random groups, grouping by domain name, and grouping by website fingerprint similarity (we evaluate their relative effects on our accuracy in §5).

We model the user as starting on a webpage chosen at random. The user then visits another webpage. With probability p_{switch}, the user switches to a different group, chosen at random, and then chooses a random webpage within that group. Otherwise, the user stays in the same group, and chooses a different page within that group at random. Low values of p_{switch} represent users who stay on one topic over the course of a browsing session, while high values of p_{switch} represent users who jump from one topic to another.

Our attack does not require that the attacker know what a user's p_{switch} is; it only needs to know the grouping of related webpages, and the reasonable assumption that, once a user visits a webpage on a particular topic, they are more likely to stay on that topic than to switch to another, for a while anyway.

Potential future extensions Our attack does not preclude—and would likely benefit from—more sophisticated models of user behavior. For example, rather than partitioning webpages into non-overlapping groups, another alternative would be to allow overlaps, e.g., for webpages spanning multiple topic areas. Another extension would be to have notions of relatedness across different groups; one might partition "news"-related sites from "politics"-related ones, but the conditional probability of going to a politics site from a news site may be higher than switching to, say, a travel site. Yet another extension would be to assign higher probabilities of switching to a given topic if the user had recently visited that topic. Developing more refined user models is left for future work, and is beyond the scope of this paper.

3.3 Attack design

Our attack comprises three broad steps:

Step 1: Apply standard website fingerprinting techniques Consider a user who visits an ordered sequence of webpages $(w_1, w_2, \ldots)$

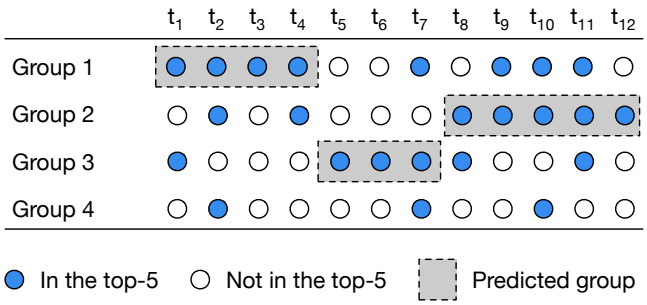


Figure 2: An example of our attack identifying the group with the longest consecutive sequence of webpages in the top-5.

using Tor, and an attacker who collects the corresponding packet traces $(t_1, t_2, \ldots)$. The attack begins by first applying any existing website fingerprinting (WF) algorithm to each t_i independently. The WF algorithm will output probabilities p_i^j: the probability that webpage w_i corresponds to webpage t_j. Top-$k(t_j)$ is the set of webpages with the k highest probabilities of corresponding to t_j.

So far, this is just the standard approach to WF. To date, WF techniques have been evaluated on their top-1 accuracy (when $w_i \in$ Top-1(t_i)) or their top-5 accuracy (when $w_i \in$ Top-5(t_i)), treating each webpage independent of one another.

Step 2: Find the longest consecutively visited group Our user model assigns each webpage w_i to a specific group $G(w_i)$. Thus, in the next step, the attacker identifies, for each packet trace t_i, the group of each webpage in Top-5(t_i). For each t_i, it computes which group has the longest consecutive sequence of appearing in the top-5, among the groups that are in Top-5(t_i). It breaks ties with the sum of the probabilities of the guessed groups.

Figure 2 provides a simplified example with four groups and a browser session spanning 12 separate webpage visits. Blue dots represent times when the respective group appeared in the top-5. Note that the attack attributes t_8 to Group 2 instead of Group 3 because Group 2 has a consecutive sequence of five pages while Group 3 has four.

Step 3: Choose webpages from the selected groups The final step of the attack is to choose, from within each selected group, the webpage with the highest probability according to the WF algorithm. For example, recall from Figure 1 that the last packet trace had two candidate webpages from the same group, and the one with the highest probability was chosen.

Using our above notation, our attack chooses

$$w_j = \text{argmax}_{w_k}\{p_k^i \mid G(w_k) \text{ is the longest sequence at } t_i\}$$

This gives us conditional probabilities, selected only from the webpages in the group that forms the longest sequence. Note in Figure 1 that the in-group webpage with the highest WF-assigned probability is not necessarily the webpage in Top-1(t_i); in fact, in practice, it often is not.

Optionally permitting gaps WF techniques do not have perfect top-5 accuracy. If the correct group had no pages in the top-5, it would appear as a *gap* in the group sequence, and in the worst case

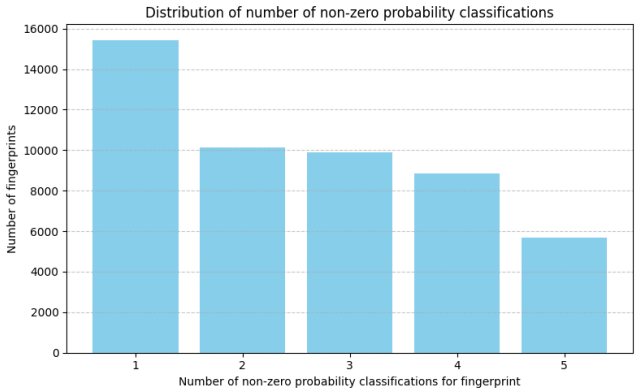


Figure 3: Distribution of number of non-zero probability classifications returned by TF.

could mean the correct group would not be identified. This can be detrimental to our attack, because it could mean that the correct group is not chosen, in which case the correct webpage will not be chosen. To mitigate this, we also consider a variant implementation in which the attacker can permit some configurable number of consecutive gaps.

Limiting to top-5 For all of our experiments, we only consider the top 5 predictions for each site in the session. To determine this top-k value, we ran 1,000 trial sessions of length 50 and recorded the number of non-zero probability classifications returned by TF per fingerprint. This result is shown in Figure 3. TF did not return more than five non-zero probability predictions for any of the trial sessions.

While TF essentially forces us to limit it to the top-5 in this paper, our technique naturally extends to other values of top-k. If a different WF technique returns more than 5 non-zero probability classifications, we can use a larger k to capture more non-zero probability/"relevant" predictions.

3.4 Limitations

Our design requires some way to determine "similarity" of webpages in order to group them. While there is no restriction imposed by our design on the exact model used to determine this, we do not provide any definitive solution in this work but simply posit that such a method of grouping exists. In our experiments we evaluate several potential methods of doing so, such as grouping pages by domain and grouping via nearest neighbors of fingerprints in a KNeighborsClassifer. Other approaches could look at text similarity metrics, utilize large language models/natural language processing, or investigate links from one webpage to another. We leave finding the optimal method to group websites to future work.

4 Fingerprint Dataset

In order to "stress-test" today's website fingerprinting techniques, we sought a fingerprinting dataset that: (1) Spanned many distinct domains, particularly those that are commonly censored by various nations, (2) Contained multiple *internal pages*, and not just the landing page of the domain, and (3) Provided the specific URL that

Dataset	Closed world size	Total domains	Samples per page	Internal pages?	Page URLs?
Wang [27]	100	100	90	No	N/A
AWF [22]	900	900	2,500	No	N/A
DF [24]	95	95	1,000	No	N/A
GE [20]	500	50	20	Yes	Yes
BE [15]	950	95	20	Yes	No
GTT23 [10]	$1.4x10^7$	$1.1x10^6$	N/A	Yes	No
Ours	7,110	466	30	Yes	Yes

Table 1: Comparison between existing datasets and our dataset. Wang, AWF, and DF were used in the original TF work.

was visited. Unfortunately, we were unable to identify any publicly available dataset that provided all three of these, so we instead collected our own.

In this section, we describe how we collected our dataset, and compare it to other publicly available ones. We then use our dataset in our subsequent analyses of our browser session fingerprinting in §5.

4.1 Data collection methodology

Domain selection We selected a total of 1,200 domains to begin building our dataset. Like in previous datasets, we decided to use the most popular sites globally as a starting point, taking the top 600 domains from the Tranco list [13] generated on 03 March 2024. We then took additional sites from the Citizen Lab test lists [12] curated specifically for discovering website censorship to model real-world usage of these attacks. From test lists for China, Iran, Russia, Saudi Arabia, and Sudan we chose 600 additional domains (120 from each list) for a total of 1,200 unique domains to select internal pages from for fingerprinting.

Internal page selection For each domain, we selected 20 pages for fingerprinting. However, the process for selecting such pages proved to be a roadblock: how do we pick internal pages from each domain that are "interesting" (likely for a user to visit)? Manually visiting each of the 1,200 domains and putting together a list of links would be too time-consuming, and scraping links from the landing page of each site didn't provide any obvious way to select "interesting" pages. The methodology used for the Hispar top web pages dataset [2] was to use the "site:" search operator in Google search, but we found that the top pages returned by Google were often unrepresentative of pages users would commonly visit. However, the same search operator in Bing provided much more "interesting" results, so we filtered for HTML results only and used a web scraper built on Python's requests and BeautifulSoup libraries to scrape all internal pages returned in the first five pages of Bing results. Domains with fewer than 20 pages were excluded from our list of pages to fingerprint. The remaining domains from the Tranco list were manually checked against the now-discontinued Alexa top sites list [23]. From the list of Tranco domains not in the Alexa list, pages that either serve as hosting domains or are unlikely to be visited by humans were also removed. This left us with 1,038 domains and 61,616 total webpages for building a dataset.

We then selected 10,000 webpages for fingerprinting - 20 pages including landing page from 500 domains. These 500 domains were made of the top 250 domains remaining from the Tranco list, and 50 random domains from each of the five Citizen Lab test lists. Internal pages used for each domain were simply the top results returned by Bing. This list of webpages served as our starting point for fingerprinting.

Crawling methodology Our fingerprinting script runs a Tor Browser in a virtual frame buffer using Xvfb, Selenium, and the Stem Python controller library. The script is contained within a Docker container to isolate from external traffic, and a subprocess running tshark sniffs specifically for the Tor traffic being sent to the guard nodes of our circuit, saving pertinent information such as timestamps, source and destination IP addresses, source and destination ports, and TLS record length. These fields are important for processing the sniffed traffic into fingerprints, as the IP addresses tell us the direction of each packet (request or response) and the TLS record length gives us the number of Tor cells sent or received. We ignore packets without a TLS record length, as the record length gives us a precise count of cells in the data stream since each cell is 514 bytes. We don't save the packet captures themselves as they consume too much storage space, and since the payloads of the packets are encrypted, we can't perform any analysis on them anyway. Additionally, caching is disabled to ensure each page load is fresh, and various datapoints like screenshots for checking for captchas, document body scroll height, start/stop timestamps for each page load, and error logs are also saved. To prevent traffic from one page load leaking into the packet capture for another, we always open a new tab for every page load and close the tab once it finishes or times out, as tshark shows a clean cutoff shortly after the tab is closed. As a result, a short sleep is added after closing the tab to allow remaining traffic to flow in before starting the next iteration.

For building our dataset, we load each page 125 times and aim for 100 valid fingerprints, which gives a buffer of 25 fingerprints for timeouts, errors, or other problems with individual fingerprint attempts – these can be checked for through the error logs. We then parallelize the data collection by running multiple containers, each running the script on different sites. These fingerprints were collected from October 2024 through March 2025.

Data validation Of course, not all fingerprints we collected were ultimately valid fingerprints. We applied a variety of filters on the initial collected set to get down to a valid set of fingerprints. Our original goal was to filter out sites with less than 100 valid fingerprints, but due to difficulties in training such a large set we decided to settle for a minimum of 30 valid fingerprints instead. A fingerprint is considered invalid if either an error was logged

Dataset	Site		Domain	
	top-1	top-5	top-1	top-5
GE	0.46	0.75	0.98	0.99
BE	0.64	0.81	0.86	0.93
Ours	0.58	0.77	0.90	0.96

Table 2: Top-1 and top-5 accuracies of both site and domain predictions for TF on GE, BE, and our dataset.

for the fingerprint or the logged scroll height for the page was zero. Additionally, if we were unable to capture a screenshot for a page we automatically rule all fingerprints for the page invalid, as a screenshot is attempted after every successful page load. And lastly, we check for fingerprints that are too short by taking the mean of the valid fingerprints and removing those that are less than half of the mean. If a site still has at least 30 fingerprints after these filters, we add 30 fingerprints for the site to the final dataset.

4.2 Dataset properties

From the fingerprinted 10,000 sites across 500 domains, we filtered down the final dataset to 7,110 unique sites across 466 domains, with a mean of 15.2 sites per domain included. Each site had 30 fingerprints associated with it. Table 1 provides a comparison of our dataset with those used in prior work.

4.3 Baseline evaluation with TF

Before evaluating our attack, we first performed a baseline evaluation on a TF model trained on our dataset and compared with models trained on the GE and BE datasets. To be consistent with GE we used a 4:1 ratio for splitting the dataset into training and testing. This split was done for each webpage across all datasets to ensure TF is not trained on the same webpages used for testing. Before training, all fingerprints were preprocessed to match the inputs for the original TF model. Each fingerprint was converted into a list of +1 and -1 values for incoming and outgoing connections, respectively. Because the original TF model requires an input with a specific size, all fingerprints were either truncated or padded with zeros to reach a length of 5,000. For each dataset, we ran 1,000 trials of length 50 and recorded the top-1 and top-5 accuracies (whether the true site is in the top-1 and top-5 predictions, respectively), as well as the top-1 and top-5 domain accuracies (whether at least the correct domain is in the top-1 and top-5 predictions, respectively). The results are shown in Table 2. We observe that even though TF can fairly reliably predict the domain of a fingerprint, it is unable to predict the specific webpage with similar levels of accuracy. Additionally, the difference between top-1 and top-5 site accuracy highlights the potential for our attack to bridge the gap and give better predictions than TF alone.

5 Results

In this section, we evaluate the accuracy of our browsing session fingerprinting attack, both with a state-of-the-art ML-based WF technique as its building block, as well as with a set of simulated WF techniques.

Throughout this section, we evaluate our attack's *accuracy* in terms of the number of webpages it was able to precisely guess.

5.1 Attack Evaluation on TF

Recall that our attack uses an existing WF technique as its building block. We use Triplet Fingerprinting [25], and evaluate all of the critical parameters of our attack.

Evaluating the effect of p_{switch} The p_{switch} parameter models how likely a user is to switch from one group (e.g., topic) of webpages to another. To evaluate its effect on our attack's accuracy, we vary it between zero and one in increments of 0.1—when p_{switch} is zero, a user never leaves the first webpage group they start in; when p_{switch} is one, they randomly jump across groups with every page visit. For each p_{switch} interval we ran 1000 trials of length 50 and recorded the accuracy. Recall that our attack does not know nor try to estimate p_{switch}; it merely looks for the group with the longest consecutive sequence of appearing in the top-5.

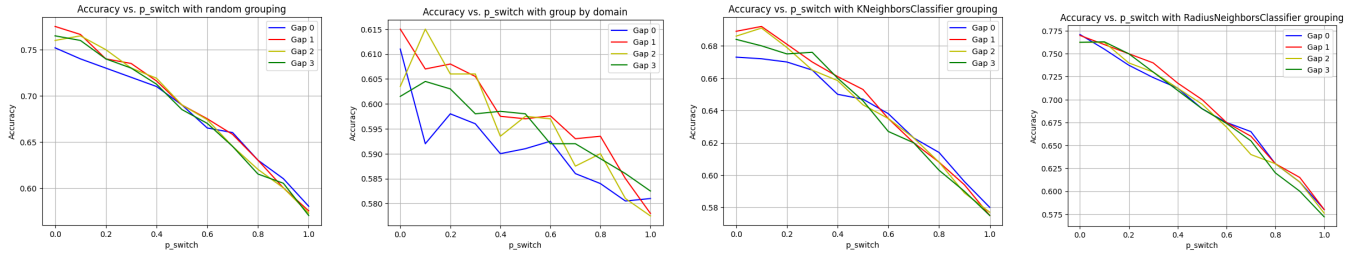
Figure 4 shows the accuracy of our attack (the y-axis) as p_{switch} (the x-axis) across various other selections of parameters we will discuss next. Across all of these results, the trend is clear: our accuracy improves as users are less likely to switch to different webpage groups. This is a natural and expected result; the longer a user stays within a group, the more chances we have to identify the group. However, a pleasantly surprising result is that our attack's accuracy smoothly transitions between the underlying model's top-1 accuracy (when p_{switch} approaches 1.0) and its top-5 accuracy (when p_{switch} approaches 0.0).

While it seems natural that our attack would reduce to standard website fingerprinting techniques' top-1 accuracy when $p_{\text{switch}} = 1.0$—after all, at that point, each webpage visit is done nearly independent of the others—it is actually possible for our attack to perform worse. This is because, by random chance, there may be a group that appears multiple times in the top-5, leading our attack to mistakenly believe that multiple pages in that group have been visited. That said, this rarely happened in our experiments.

Evaluating the effect of the grouping method Another critical parameter in our attack is how the websites are grouped together. A sophisticated model of user behavior would account for this, but that is beyond the scope of this paper, so instead we evaluate several different groupings that we believe approach worst-case settings.

The groupings we evaluated are:

(1) *Random grouping*: We split all of the webpages up into groups of size 10, with each webpage being placed into a random group.
(2) *Domain grouping*: We placed all of the pages from a given domain in the same group, allowing variable group sizes as large as 20.
(3) *K-neighbors classifier grouping*: TF converts each fingerprint into a vector in a high-dimensional Euclidean space. This method groups together the ten webpages whose vectorized fingerprints are closest together. We did this in the hopes of creating a worst-case scenario, in which it would be difficult to accurately classify pages within a given group.
(4) *Radius Neighbors Classifier Grouping*: Like with K-neighbors, this technique also groups nearby vectors together, but by



(a) Accuracy given random groups of size 10. **(b) Accuracy given groups formed by domain.** **(c) Accuracy given on groups of size 10 formed by KNeighborsClassifer.** **(d) Accuracy given groups of size 10 formed by RadiusNeighborsClassifier with radius 0.6.**

Figure 4: Attack accuracy at each p_{switch} interval for each grouping method and gap length.

specifying a small radius of 0.6 within which to gather nearby vectors.

Figure 4 shows that different groupings can have an effect on our attack's accuracy. Our worst-performing grouping (group by domain) offers only a slight improvement over the baseline, while other groupings such as random grouping and Radius Neighbors Classifier grouping can achieve close to the baseline top-5 site accuracy for low p_{switch} values.

These results are encouraging, as they show that even in contrived, worst-case scenarios, our attack still has a marked improvement over standard WF attacks.

Evaluating the effect of the max gap lengths Recall that our attack can optionally allow for "gaps" in the the contiguous sequence of groups in the top-5. We evaluate several different max gap sizes: 0 (no gaps allowed), 1, 2, and 3. For this experiment, we used random groupings of webpages.

We observe that allowing for gaps in the contiguous sequence can offer a slight increase in performance as well, particularly at low p_{switch} values. This is surprising, as TF does not provide nonzero probabilities outside of the top-5. A WF technique that would give such probabilities would likely see an increased utility in gaps.

Evaluating effect of group size One other parameter that is not tested in the previously mentioned experiments is group size. Here we evaluate the effects of group size on the attack's performance. To evaluate this, we used random grouping with group sizes of [1, 2, 5, 10, 30, 50, 100, 200, 400] and no gap. For each group size, we ran 1000 trials of length 50 at each increment of p_{switch} and recorded the accuracy. These results are shown in Figure 5. We observe that group sizes at the upper and lower bounds have lower accuracies, and the best accuracies in the middle come close to the baseline top-5 site accuracy of 0.77. Note that this attack with group size 1 is the same as just using the baseline TF.

Evaluation on different datasets Lastly, we compare our attack's performance on three datasets: ours, BE, and GE. For this, we used random grouping with group size 10, no gap, and the same p_{switch} increments as previous experiments. For each combination of parameters and dataset, we ran 1000 trials each of length 50 and recorded the accuracy. The results are shown in Figure 6. These results show that our attack is still able to improve upon TF's baseline performance even when other datasets are used instead of ours. The

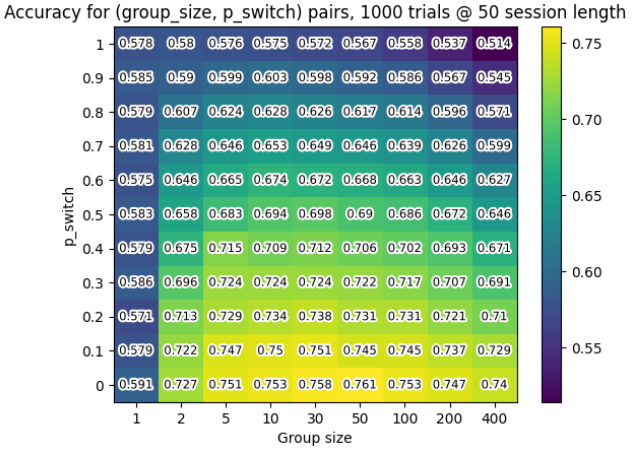


Figure 5: Accuracy given pairings of group size and p_{switch} on random grouping.

improvement also follows a fairly consistent curve: at low p_{switch} values we are close to baseline top-5 accuracy for each dataset, and we fall off to baseline top-1 accuracy once p_{switch} has a high value.

Collectively, these results show the efficacy of our technique on improving TF, particularly if the user does not switch groups frequently, where we can achieve close to baseline top-5 accuracy. This improvement is observed under a variety of scenarios - different groupings, different group sizes, and different datasets used.

5.2 How does it generalize to other fingerprinters?

Our evaluation thus far has involved use of the state of the art Triplet Fingerprinting technique. A natural question to ask is: how well does our technique generalize to other fingerprinters? Does it improve the performance of WF techniques that have markedly *bad* top-1 and top-5 accuracy? Will the attack still be beneficial if future WF techniques significantly improve top-1 accuracy?

Simulator design To evaluate this, we simulated fingerprinters with a wide range of accuracies. Two parameters were given for the simulator to simulate error: a top-1 accuracy parameter, which set

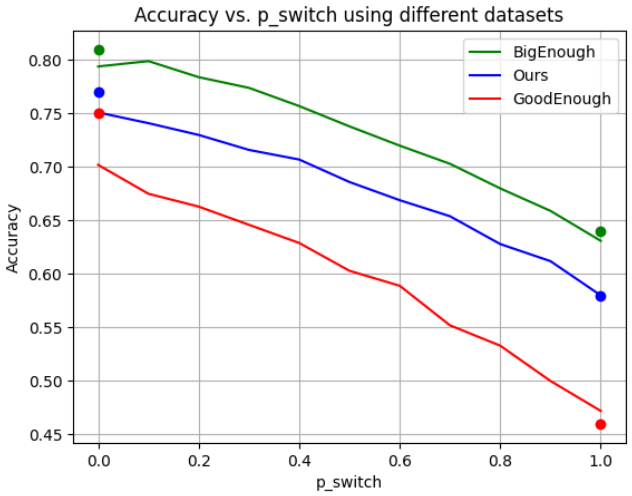


Figure 6: Accuracy comparison for the attack on our dataset, BE, and GE. The dots on the left denote TF's top-5 accuracy, and the dots on the right denote TF's top-1 accuracy for the respective datasets (from Table 2).

the probability that the simulator would have the correct site as the top prediction, and a top-5 parameter, which set the probability that the correct site would be in the top-5 predictions *given that it is not in the top-1*.[2] For all sites not in the top-5, we give a probability of 0, and the rest of the top-5 is filled with random sites with decreasing probability relative to their rank. To run, the simulator simply takes in the list of test session sites and based on the top-1 and top-5 accuracy parameters chooses where to place the correct "prediction," returning the simulated top-k predictions for each site in the list.

Results We evaluate the performance of our attack on the simulated predictions. We have a set of 950 "sites" (same as BE) to serve as our "fingerprints." For this experiment we generate random groups of size 10. There are three parameters we are changing here: top-1 accuracy, top-5 accuracy, and p_{switch}. Each is tested at increments of 0.1 in the range of [0,1]. For each combination of (top-1 accuracy, top-5 accuracy, p_{switch}) we run 100 trials each of length 25 and record both the accuracy of our attack and the difference from the simulator's top-1 accuracy. For these experiments, we set the maximum gap size to zero.

The accuracy results for p_{switch} =0.2, 0.5, and 0.8 are shown in Figures 7; Figure 8 shows how much these accuracy results improve (or diminish) relative to the simulated fingerprinter's top-1 accuracy. From these results we see that the key factor appears to be the top-5 accuracy of the fingerprinter—if it is reasonably good at getting the true site in the top-5, then the attack performs well even if the true site is not frequently in top-1. This presents an interesting path forward for future WF techniques: combined with our attack, top-5 accuracy may suffice. On the other hand, we observe that if the fingerprinter already has an extremely high top-1 accuracy, our attack often ends up hurting more than it helps.

[2]If α = the top-1 accuracy and β = this top-5 accuracy, then the traditional top-5 accuracy is $\alpha + (1 - \alpha) \cdot \beta$.

6 Conclusion

Users rarely open a browser, visit a single webpage, and then shut their browser. Likewise, users rarely visit a series of websites chosen uniformly at random, with no relevance to one another. And yet, to date, website fingerprinting attacks have effectively assumed these.

In this paper, we show that taking into account not just the immediate webpage a user is visiting but rather their entire browsing session can help provide contextual clues that can significantly improve website fingerprinting accuracy. We presented our attack, which uses existing WF techniques as a building block, along with novel processing that refines and improves the WF tools' findings. We evaluated it on existing fingerprint datasets as well as a new one we collected for this work. To evaluate how our attack generalizes to other, future WF techniques, we also simulated them with widely varying degrees of accuracy. Collectively, our results show that our attack improves WF accuracy, effectively boosting their accuracy from top-1 rates to top-5 rates.

There remain several limitations, areas of improvement, and interesting avenues of future work. Chief among these is the need to study, develop, and analyze accurate models of user browsing behavior. The results in this paper show that changes in behavior (the value of p_{switch}, how large groups of related webpages are, and how they are grouped) can have significant impact on the results of this attack. Fortunately, in most cases, our attack still improves the performance of top-1 WF.

We believe that all future WF work should evaluate browsing sessions, rather than singular webpages. To support these efforts, we will be making our code and data publicly available.

References

[1] Mashael AlSabah, Kevin Bauer, Tariq Elahi, and Ian Goldberg. 2013. The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting. In *Privacy Enhancing Technologies Symposium (PETS)*.

[2] Waqar Aqeel, Balakrishnan Chandrasekaran, Bruce Maggs, and Anja Feldmann. 2020. On Landing and Internal Pages: The Strange Case of Jekyll and Hyde in Internet Measurement. In *ACM Internet Measurement Conference (IMC)*.

[3] Arushi Arora and Christina Garman. 2025. Improving the Performance and Security of Tor's Onion Services. In *Privacy Enhancing Technologies Symposium (PETS)*.

[4] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *ACM Conference on Computer and Communications Security (CCS)*.

[5] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security (CCS)*.

[6] David L. Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* 24, 2 (1981).

[7] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. 2022. Online Website Fingerprinting: Evaluating Website Fingerprinting Attacks on Tor in the Real World. In *USENIX Security Symposium*.

[8] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*.

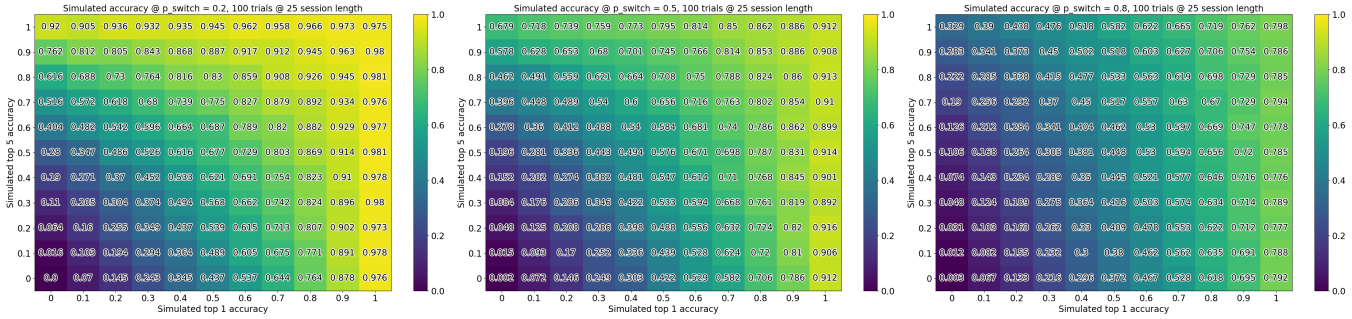
[9] Gregory Fleischer. 2009. Attacking Tor at the Application Layer. (2009). `https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-gregory_fleischer-attacking_tor.pdf` Defcon.

[10] Rob Jansen, Ryan Wails, and Aaron Johnson. 2024. *A Measurement of Genuine Tor Traces for Realistic Website Fingerprinting*. Technical Report. doi:`10.48550/arXiv.2404.07892`

[11] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. 2015. Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services. In *USENIX Security Symposium*.

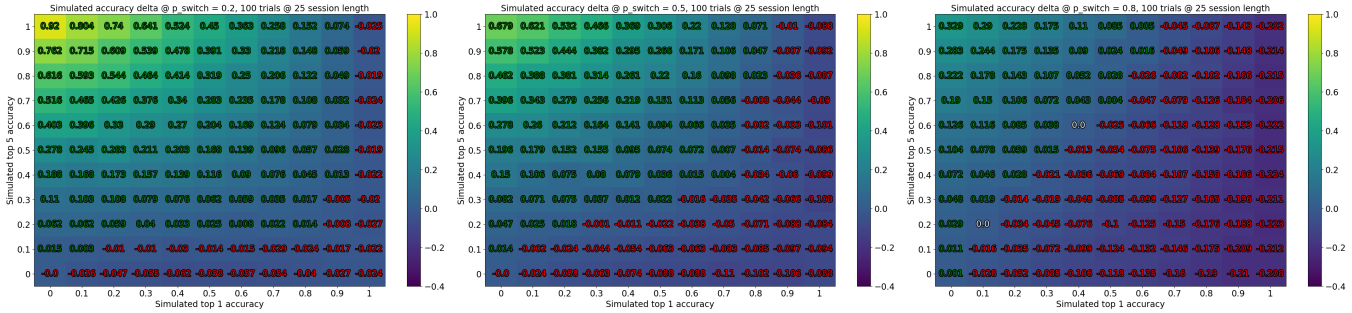
[12] Citizen Lab. [n. d.]. Block test list. `https://github.com/citizenlab/test-lists`.

[13] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites



(a) p_{switch} =0.2 (b) p_{switch} =0.5 (c) p_{switch} =0.8

Figure 7: Attack accuracy given simulated predictions on random groups of size 10 with no gap.



(a) p_{switch} =0.2 (b) p_{switch} =0.5 (c) p_{switch} =0.8

Figure 8: Attack accuracy difference from simulated baseline predictions on random groups of size 10 with no gap. Green text is where we improve, red is where we do worse, and white is equal.

Ranking Hardened Against Manipulation. In *Network and Distributed System Security Symposium (NDSS)*.

[14] Zhihao Li, Stephen Herwig, and Dave Levin. 2017. DeTor: Provably Avoiding Geographic Regions in Tor. In *USENIX Security Symposium*.

[15] Nate Mathews, James K Holland, Se Eun Oh, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. 2023. SoK: A Critical Evaluation of Efficient Website Fingerprinting Defenses. In *IEEE Symposium on Security and Privacy*.

[16] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *ACM Conference on Computer and Communications Security (CCS)*.

[17] Lasse Øverlier and Paul Syverson. 2007. Improving efficiency and simplicity of Tor circuit establishment and hidden services. In *International Workshop on Privacy Enhancing Technologies*.

[18] Andriy Panchenko, Asya Mitseva, Martin Henze, Fabian Lanze, and Klaus Wehrle. 2017. Analysis of Fingerprinting Techniques for Tor Hidden Services. In *Workshop on Privacy in the Electronic Society (WPES)*.

[19] Muhammad Talha Paracha, Balakrishnan Chandrasekara, David Choffnes, and Dave Levin. 2020. A Deeper Look at Web Content Availability and Consistency over HTTP/S.

[20] Tobias Pulls. 2020. *Towards Effective and Efficient Padding Machines for Tor.* Technical Report. doi:10.48550/arXiv.2011.13471

[21] Michael Reininger, Arushi Arora, Stephen Herwig, Nicholas Francino, Jayson Hurst, Christina Garman, and Dave Levin. 2021. Bento: Safely Bringing Network Function Virtualization to Tor. In *ACM SIGCOMM*.

[22] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. 2018. Automated Website Fingerprinting through Deep Learning. In *Network and Distributed System Security Symposium (NDSS)*.

[23] Ali Sawyer. 2011. alexa-top-1m. `https://github.com/ali-sawyer/alexa-top-1m`.

[24] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *ACM Conference on Computer and Communications Security (CCS)*.

[25] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. 2019. Triplet Fingerprinting: More Practical and Portable Website Fingerprinting with N-shot Learning. In *ACM Conference on Computer and Communications Security (CCS)*.

[26] Tor Metrics – Users 2025. Tor Metrics – Users. `https://metrics.torproject.org/userstats-relay-country.html`.

[27] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium*.

[28] Tao Wang and Ian Goldberg. 2013. Improved Website Fingerprinting in Tor. In *Workshop on Privacy in the Electronic Society (WPES)*.

[29] Tao Wang and Ian Goldberg. 2017. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *USENIX Security Symposium*.

[30] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. 2012. StegoTorus: A Camouflage Proxy for the Tor Anonymity System. In *ACM Conference on Computer and Communications Security (CCS)*.

[31] Philipp Winter and Jedidiah R. Crandall. 2012. The Great Firewall of China: How It Blocks Tor and Why It Is Hard to Pinpoint. *;login:* 37, 6 (2012), 42–50.